Bitcoin Price Prediction Using LSTM Model

A Project Report Submitted in the
Partial Fulfillment of the Requirements
for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

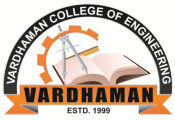
Submitted by

Nikhil Dilip Dhore	17881A05L9
V Bhanu Prakash	17881A05J8
K Abhinav Reddy	17881A05J3

SUPERVISOR
B. Suresh Kumar
Assistant Professor

Department of Computer Science and Engineering
VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD
An Autonomous Institute, Affiliated to JNTUH

May, 2021



VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD

An Autonomous Institute, Affiliated to JNTUH

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project titled **Bitcoin Price Prediction Using LSTM Model** is carried out by

Nikhil Dilip Dhore	**17881A05L9**
V Bhanu Prakash	**17881A05J8**
K Abhinav Reddy	**17881A05J3**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** during the year 2020-21.

Signature of the Supervisor
B. Suresh Kumar
Assistant Professor

Signature of the HOD
Dr. Rajanikanth Aluvalu
Professor and Head, CSE

Kacharam (V), Shamshabad (M), Ranga Reddy (Dist.)–501218, Hyderabad, T.S.
Ph: 08413-253335, 253201, Fax: 08413-253482, www.vardhaman.org

Acknowledgement

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **B. Suresh Kumar**, Assistant Professor and Project Supervisor, Department of Computer Science and Engineering, Vardhaman College of Engineering, for his able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Dr. Rajanikanth Aluvalu**, the Head of the Department, Department of Computer Science and Engineering, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R. Ravindra**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Computer Science and Engineering department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

Nikhil Dilip Dhore

V Bhanu Prakash

K Abhinav Reddy

Abstract

Bitcoin is one of the first cryptocurrency to be introduced in the global markets in the year 2008. It is one of the many methods used by investors for the purpose of investment. It is a digital currency whose price has been rising in past few years, and sometimes has seen sudden decline in it without knowing any reason behind it. Because of these unpredictable fluctuations in its price, it is very risky for anyone who wants to invest in it. In this research study we are building a model for predicting bitcoin price with the help of LSTM (Long Short Term Memory) module. This module is advancement to the RNN which was later developed and popularized by many researchers. Similar to RNN, LSTM module includes various modules for providing the recurrent consistency. The methods that we apply in this system, and also the techniques and tools to predict Bitcoin price includes pandas, Keras, numpy and Tensorflow. Using these tools or packages we build a model that helps in predicting the future price of the bitcoin which can help the traders to maximize their profits. This is a type of time-series analysis based on the data derived from the past. We have gathered the data from Yahoo Finance India. The dataset consists of 7 attributes and the number of instances available are 2440. The accuracy obtained in this study was calculated around 85%. In this study we are also trying to forecast the price pattern for the next 20 days.

Keywords: Deep Learning, Recurrent Neural Networks, Long Short Term Memory, Comma Separated Values, Auto Regressive Integrated Moving Average, Bitcoin, Time-series, Cryptocurrency.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Description
DL	Deep Learning
RNN	Recurrent Neural Networks
ANN	Artificial Neural Networks
SVM	Support Vector Machine
ARIMA	Auto Regressive Integrated Moving Average
LSTM	Long Short Term Memory
UML	Unified Modeling Language
SRS	Software Requirement Specification
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
UI	User Interface

CHAPTER 1

INTRODUCTION

1.1 Motivation

In past few years cryptocurrency has taken the world by storm, Bitcoin is one of its which was first introduced in the year 2008, but was formally mined on 03 January 2009[1]. This was known as the Genesis block of bitcoin also called as the Block 0.The person who mined this block was Satoshi Nakamoto. As bitcoin is one of the first cryptocurrency to become accepted world-wide it is also considered as the originator of other cryptocurrencies. Crypto's are not monitored or controlled by any government authorities or agencies. These are purely P2P network which makes it difficult to track and hence it leads to reduction of scams and frauds. It is one of the most accepted form of digital currency for transactions[2][3]. Many organizations and retailers accept it as medium of exchange. The initial years of bitcoin did not fascinate many people, they were reluctant to invest in it. As people became more and more aware of this digital currency they started investing in it, as a result there was huge fluctuation in bitcoin price. On 20th November 2017 the bitcoin price was around $8,100; in the next 26 days it saw a huge surge in its price, on 15th December 2017, bitcoin was on all-time high of $17,900 [4][3]. This was first time that there was huge fluctuation in the price. But later on it again fell by around 50%. Later on, due to COVID-19 pandemic in the year 2020 bitcoin started gaining momentum and has increased in such a way that everyone were astonished by its growth. It reached at record breaking high of $60,000 in April 2021 [5]. Since, the price pattern of bitcoin is very difficult to predict and understand we are coming up with one of the concepts in deep learning [6] which can be used to understand the price pattern and help people in gaining insight on bitcoin prices.

1.2 Problem Definition

As discussed in the above section, bitcoin price fluctuations are very unpredictable hence, this increases the risk of investment. Many people have started gaining knowledge about the bitcoin, and how the entire process works, due to this understanding they are attracted to invest in it. But, while investing there is a huge risk that they are ready to take, a risk in which they can loose all their money or can make huge profits. In this study we are trying to solve the question that many investors have regarding the growth of the bitcoin. We will be predicting the price pattern associated with bitcoin depending on the historical data.

1.3 Objectives of Project

These are the objectives that are associated with our bitcoin price analysis:

- To develop a RNN model to predict the Bitcoin price.
- To help traders in learning the price pattern of Bitcoin.
- To apply machine learning algorithms to do time series analysis.

1.4 Organization of Documentation

In this section we will be looking at the overview of the entire report. This tells about the detailed description of each chapter.The various chapters and their details are listed below.

- Chapter 1: Introduction

  This chapter contains the details about the motivation behind the project, the problem definition and the objectives.

- Chapter 2: Literature survey

  In this chapter we have discussed the literature survey done for the

project, and what are the various drawbacks that we noticed in other models.We have also discussed our proposed system.

- Chapter 3: Analysis

  In this chapter we have gathered the user, software and hardware requirements for developing the system. We also describe the architecture of our system along with the various modules used in it.

- Chapter 4: Design

  This is the chapter where we have discussed the designing of the system. It contains various UML diagrams.

- Chapter 5: Implementation and Result Analysis

  The implementation phase and the result analysis is described in this chapter.

- Chapter 6: Conclusion

  Here the conclusion and future enhancements are discussed briefly.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Bitcoin is the largest cryptocurrency and also the first one to be introduced in the global markets. Initially when it was introduced, it had almost no existence, the value was very less. It was been traded at $0.0008 to $0.08 per coin in July 2010 [7]. But, as the years passed and people gained knowledge about it, it's price sky rocketed and reached all time highs. Later, it saw many ups and downs and due to these unpredictable fluctuations in it's prices there is a need for an system that predicts the future price patterns, so that people can invest by taking minimum risks.

2.2 Existing System

Bitcoin is the biggest cryptocurrency with the market cap of around $814.7 billion as dated on 17th May 2021 [5]. Bitcoin is decentralized and not controlled by any individual [1]. The details regarding the bitcoin are stored in public ledgers called Blockchain [1][8]. Since, this public and also very secure many people are attracted towards this, as a result the price fluctuations are very high [9]. There were many research studies conducted inorder to gain insight on the price pattern of bitcoin are many were able to gain satisfying results. The models which were used for these researches included ARIMA, ANN [10], RNN [11] (Recurrent Neural Networks), SVM [12] (Support Vector Machines). The results gained from these models were close to the actual price but had their own disadvantages.

2.3 Disadvantage of Existing systems

The disadvantages of various prediction models are discussed below:

I. ARIMA:

1. This works well with small datasets, but when it comes to larger datasets it is not efficient.
2. There are multiple parameters (p, q, d) which must be calculated based on the data.
3. It performs well when the data is stationary, i.e. when there no seasonality, trend etc.

II. ANN:

1. There is high dependence on the hardware and architecture.
2. There is problem of unexplained behaviour of the network.
3. It is difficult to determine the proper network structure.

III. RNN:

1. The computation cost of these networks is very high.
2. Training is a very tedious and difficult process.
3. Determination of activation functions become hard for long sequences.
4. It faces issues like Exploding or Vanishing Gradient.

IV. SVM:

1. This is not suitable for larger datasets.
2. If the dataset has lot of noise and irregularities, this model cannot predict results accurately.
3. This model won't provide accurate results if the count of features exceed the number of data samples used for training.

The above mentioned models do not work well when it comes to larger datasets, also the computation cost is a concern. Apart from these points there is also a major factor which includes vanishing gradient or exploding gradient problem. This issue makes the model less efficient as the learning capability decreases.

2.3.1 Vanishing Gradient Problem

Vanishing Gradient Problem is one of the problems that is encountered while we train a neural network which depends on the gradient based methods like Back Propagation. Due to this very reason it becomes a tedious task of tuning the hyper parameters of the preceding layers. This problem affects the model even more, if the number of hidden layers increase in the network. These methods have the ability to learn and tune the hyper parameters based on the understanding of how a minor change in the parameter's value can affect the network's output. If change in the parameter's value causes very minute change to the network's output then the system cannot tune the parameter effectively.

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$$

This difficulty is known as the vanishing gradient problem. If the network's output is hardly effected by the minor changes in the parameter values, we can say that even a large change in its value doesn't have a big impact in the output.

2.3.2 Exploding Gradient Problem

The exploding gradient problem is an issue found while training an ANN model which has gradient-based learning methods like backpropagation. Exploding gradients is an issue when huge error gradients pile up and result in vast deviation to the network's weights during training. Exploding gradients may result in instability of the network that cannot be trained using the

provided training data. Which can further be affected when it comes to the large sequence of data.

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$$

Few common ways used in addressing the vanishing and exploding gradient problem are:

- Reducing the amount of Layers
- Use an activation function that doesn't reduce gradients, like ReLU
- Gradient Clipping
- Add skip connections
- Pre-training using auto encoder
- Weight Initialization

2.4 Proposed System

In our proposed system we are building a LSTM model which overcomes the vanishings gradient and exploding gradient problems. This model makes use of multiple hidden layers that consists of LSTM cells that have the ability to store the data from previous layers. Coming to the architecture, we have a dataset that contains historic data about the bitcoin prices. We have collected the data from the yahoo finance website. We then pre-process the data to remove the irregularities and noises in that data. After the pre-processing stage we, split the data into testing data and training data. We have taken 75% of data as training data and the rest 25% is being used for testing the model. We then build the model and train it using the training data and attain a highly accurate model. Further on we predict the future prices and then test the output using the testing data. At last we draw the conclusions by plotting graphs.

2.4.1 LSTM Model

LSTM is an enhancement to the traditional RNNs. These have the capability to memorize the historical data. The problems like vanishing gradient of the RNN is solved here. LSTM can be used for classifying, processing and predicting the time series for a given time lag. It trains the system with the help of back-propagation. Following are the components of a LSTM network:

A. Cell State

A cell state carry information from one time step to another. A cell consists of three gates (forget, input and output) that add information to the cell state. Adding information to cell state is dependent on the sigmoid function. 0 means add "no information", 1 means add "complete information".

$$\tilde{c_t} = tanh(w_c[h_{t-1}, x_t] + b_c)$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{c_t}$$
$$h_t = o_t * tanh(c^t)$$

B. Forget Gate

Let's say that, the previous few time steps encode the information about the gender of the subject. This is useful to predict the next few words when the subject is same. But when a new subject enters, there is no requirement of retaining memory regarding the information about the gender. This is what the forget gate gets trained to do. It concatenates the previous hidden state to the current input, multiplies it with weights, and adds a bias. Later we apply a sigmoid function before multiplying it to the cell state.

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

.

C. Input Gate

This determines which data is required to be saved to the cell state. It simply does the same operation as the forget gate, instead of writing it onto the cell state, it combines (multiplies) it with Tanh (squashed) of the concatenated vector of hidden state and input (plus bias). This is then added to cell state, that has been already updated by the forget gate.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

D. Output Gate

Finally, we decide on what is the output of the LSTM cell. This is done simply by applying a sigmoid function on the concatenation of the previous hidden state and current input. And then we multiply it with Tanh (squashed) version of the cell state, which contains the information about what is to be remembered and what needs to be forgotten.

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

where,

i_t = represents input gate
f_t = represents forget gate
o_t = represents output gate
σ = represents sigmoid function
w_x = weight for the respective gate (x) neuron
h_{t-1} = output of previous lstm block
x_t = input at current timestamp
b_x = bias for the respective gate (x)
c_t = cell state at timestamp (t)

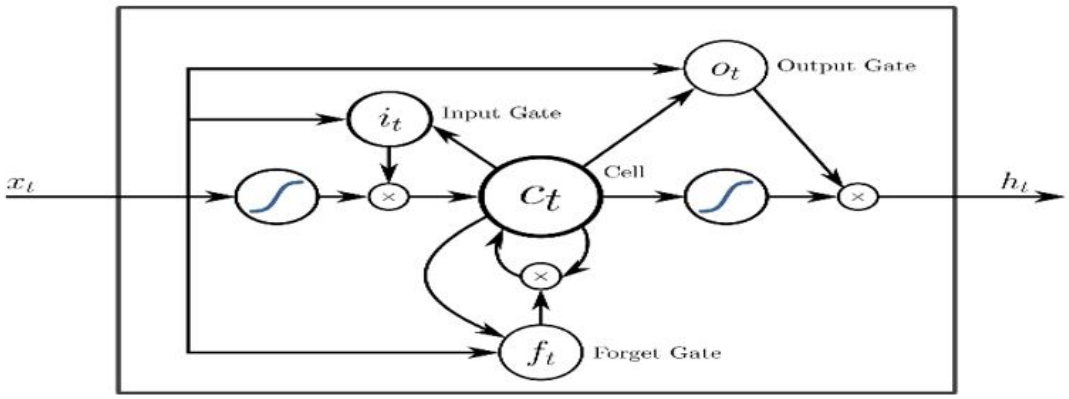


Figure 2.1: LSTM Cell

The above figure depicts the single LSTM cell architecture. It consists of input gate, output gate, forget gate and cell state.

CHAPTER 3

ANALYSIS

3.1 Introduction

Iterative Process Model

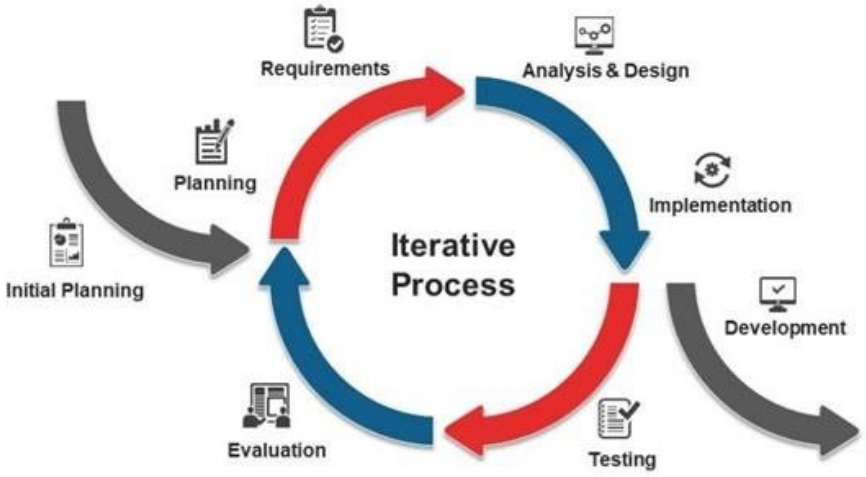


Figure 3.1: Iterative Model

In the Iterative model, iterative interaction begins with a straightforward execution of a little arrangement of the product necessities and iteratively improves the developing variants until the total framework is carried out and fit to be sent.

An iterative life cycle model doesn't endeavor to begin with a full particular of prerequisites. All things considered,advancement starts by determining and executing simply part of the product, which is then audited to recognize further necessities. This cycle is then rehashed, delivering another form of the product toward the end of every emphasis of the model. Iterative cycle begins with a straightforward execution of a subset of the product necessities and iteratively upgrades the advancing variants until the full framework is carried out. At each emphasis, plan alterations are made and new useful abilities are added. The essential thought behind this strategy is to foster a

framework through rehashed cycles (iterative) and in more modest parts all at once (steady). Iterative and Incremental advancement is a mix of both iterative plan or iterative technique and gradual form model for advancement. "During programming advancement, more than one emphasis of the product advancement cycle might be in progress simultaneously." This interaction might be depicted as an "evolutionary acquisition" or "incremental build" approach."

In this model, the entire prerequisite is isolated into different forms. During every emphasis, the advancement module goes through the prerequisites, plan, execution and testing stages. Each resulting arrival of the module adds capacity to the past discharge. The interaction proceeds till the total framework is prepared according to the necessity.

3.2 Software Requirement Specification

Software Requirement Specification (SRS) as the name proposes, is finished determination and depiction of necessities of programming that should be satisfied for fruitful improvement of programming framework. These prerequisites can be utilitarian just as non-necessities relying on sort of prerequisite. The communication between various clients and worker for hire is done on the grounds that its important to completely see needs of clients.Contingent on data assembled after connection, SRS is created which depicts prerequisites of programming that may incorporate changes and adjustments that is should have been done to build nature of item and to fulfill client's interest.

3.2.1 User requirement

Coming to the user requirements, the user who needs to know the future price pattern of bitcoin must load a file that contains the historical data of the bitcoin prices. Upon uploading the file the user can see the future pattern for 20 days which has been generated in the form of a graph.

3.2.2 Software requirement

The following are the software requirements needed in order to run this model:

- Operating System: Windows, Mac OS, Linux
- Language: Python 3.5 and above
- Tool: Jupyter Notebook
- Libraries: Tensorflow, Pandas, Numpy, Keras, Matplotlib, sklearn

3.2.3 Hardware requirement

The following are the hardware requirements needed in order to run this model:

- Processor: i3 and above
- System RAM: 8GB minimum
- Hard disk space: 128GB minimum

3.3 Architecture of Project

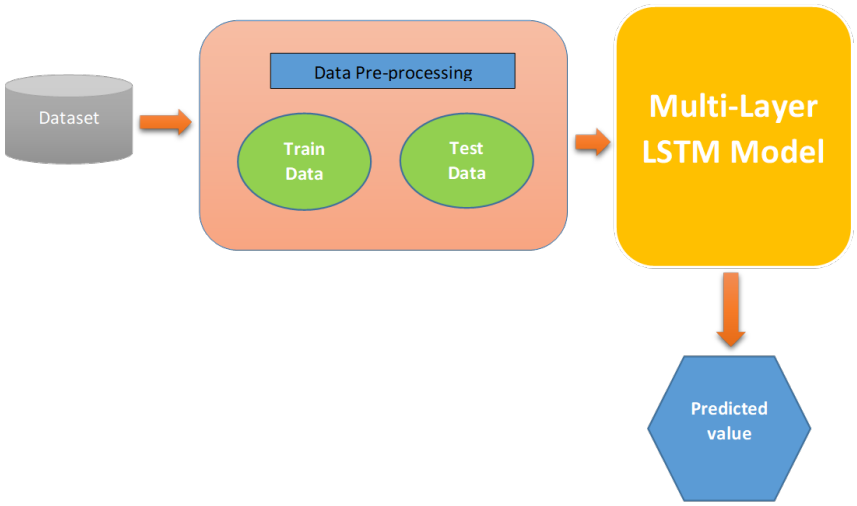


Figure 3.2: Architecture Diagram

The above figure depicts the architecture of the project. In this there is a data set which contains the historical data related to the bitcoin prices that has been collected from the yahoo finance India website. Once the data is loaded into the system, it is pre-processed to remove the irregularities and missing data. After the pre-processing stage the data is then split into training set and testing set in the ratio of 3:1. We then build a multi-layer LSTM model, which is trained using the training data. Once the training is completed, we predict the future prices. These predicted prices are compared against the test data and then the comparison graphs are plotted.

3.4 Algorithms and Methods

The model that we have used to predict the future price of bitcoin is an advancement to the traditional recurrent neural networks. The algorithm that we have implemented is given below.

1. Import all the required libraries.
2. Load the data set,which is in csv format by using the read_csv() method.
3. Now, we perform feature scaling i.e.normalize the data by using MinMaxScalar() so as to avoid over-fitting and under-fitting.
4. We split the data into training data and testing data in 3:1 ratio.
5. Now, we build a multi-layer LSTM model, in-order to initialize the neural network we use sequential model.
6. Now, to the sequential model we add LSTM layers with the dimensionality of 50 units and Dropouts to prevent over-fitting.
7. We train the above built model by using the training data. After training we store the model.
8. Now, we predict the future price by using the testing data.
9. After predicting the price, we denormalize the data so as to get actual values.
10. Finally we plot the graphs comparing the original price and the predicted price.

The above algorithm is represented in the form of a flowchart.

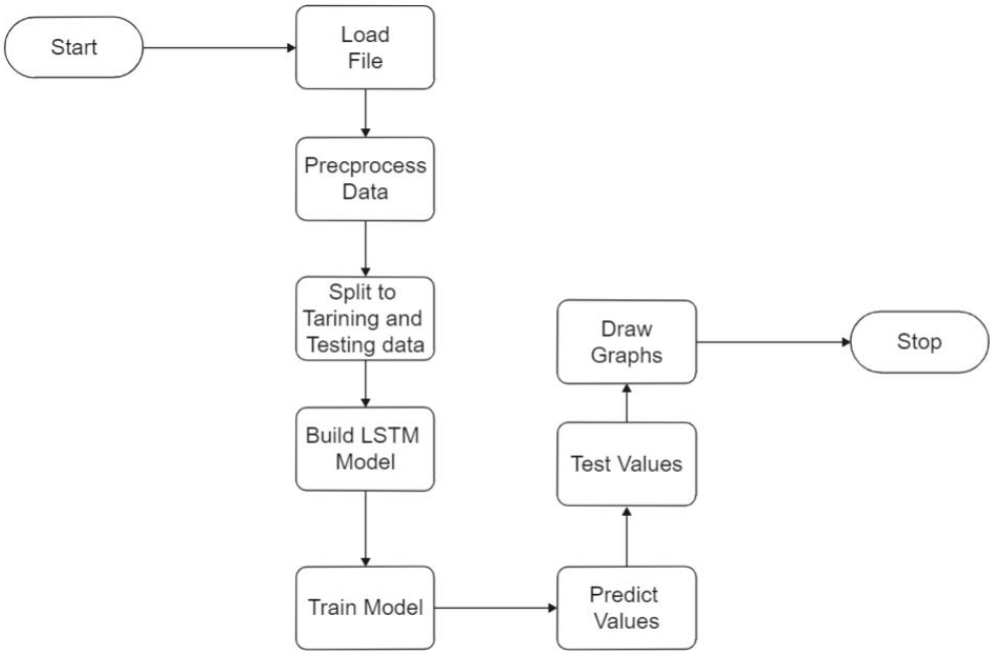


Figure 3.3: Flowchart

CHAPTER 4

DESIGN

4.1 Introduction

Designing is an important phase when it comes to application development. Without proper design we cannot build an effective and user-friendly application. A good design provides good user experience and better look and feel to the system. When it comes to designing there are two main criteria that come into picture.

- Input Design
- Output Design

Input Design:
This is a piece of generally framework plan. The primary goal during the input design is as given.

- To deliver a financially savvy strategy for input.
- To accomplish the most noteworthy conceivable degree of accuracy.
- To guarantee that the info is satisfactory and perceived by the user.

Output Design:
Yields from PC frameworks are required basically to convey the consequences of handling to clients. They are additionally used to give a lasting duplicate of the outcomes for later counsel. The different sorts of yields overall are given below.

- Outer Outputs, whose objective is outside the organization.
- Internal Outputs whose objective is inside organization.
- User's fundamental interface with the computer.

- The yields were should have been created as a printed version and just as questions to be seen on the screen. Keeping in see these yields, the configuration for the yield is taken from the yields, which are at present being acquired after manual handling. The standard printer is to be utilized as yield media for hard copies.

- Operational yields whose utilization is absolutely inside the PC department.

4.2 UML Diagrams

Unified Modeling Language (UML) is a universally useful demonstrating language. The fundamental point of UML is to demonstrate a standard method to imagine the way a framework has been planned. It is very like diagrams utilized in other fields of designing. UML isn't a programming language; it is fairly a visual language. We use UML charts to depict the conduct and design of a framework. UML helps programmers, finance managers and framework engineers with demonstrating, plan and investigation.
The are a total of 14 UML diagrams which have been classified into two categories:

Behavioural Diagrams

- Usecase Diagram
- Sequence Diagram
- Activity Diagram
- State Machine Diagram
- Communication Diagram
- Interaction Overview Diagram
- Timing Diagram

Structural Diagrams

- Class Diagram
- Collaboration Diagram
- Deployment Diagram
- Component Diagram
- Object Diagram
- Profile Diagram
- Composite Structure Diagram

4.2.1 Usecase Diagram

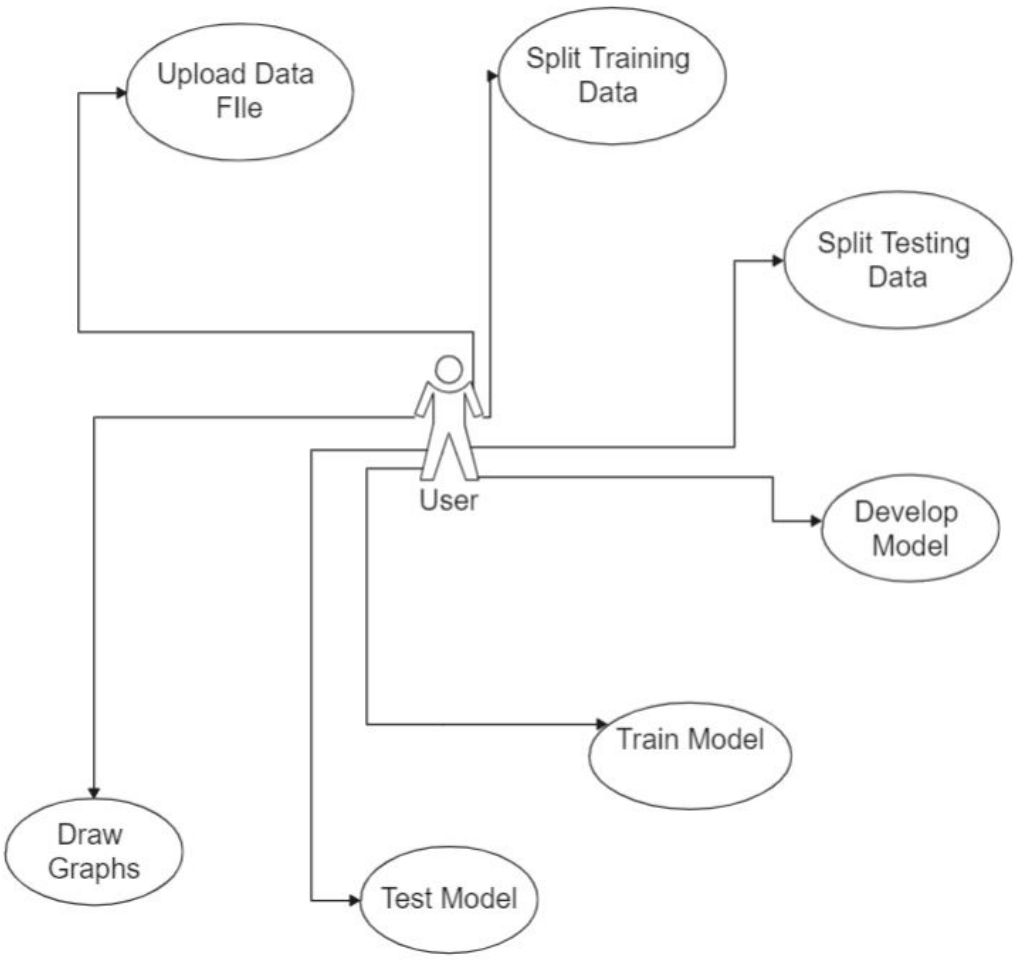


Figure 4.1: Usecase Diagram

The above usecase diagram consists of an actor (user) who can perform multiple tasks called as usecases. The various usecases of the user are, uploading data file, splitting training data, splitting testing data, develop model, train model, test model and draw graphs.

4.2.2 Sequence Diagram

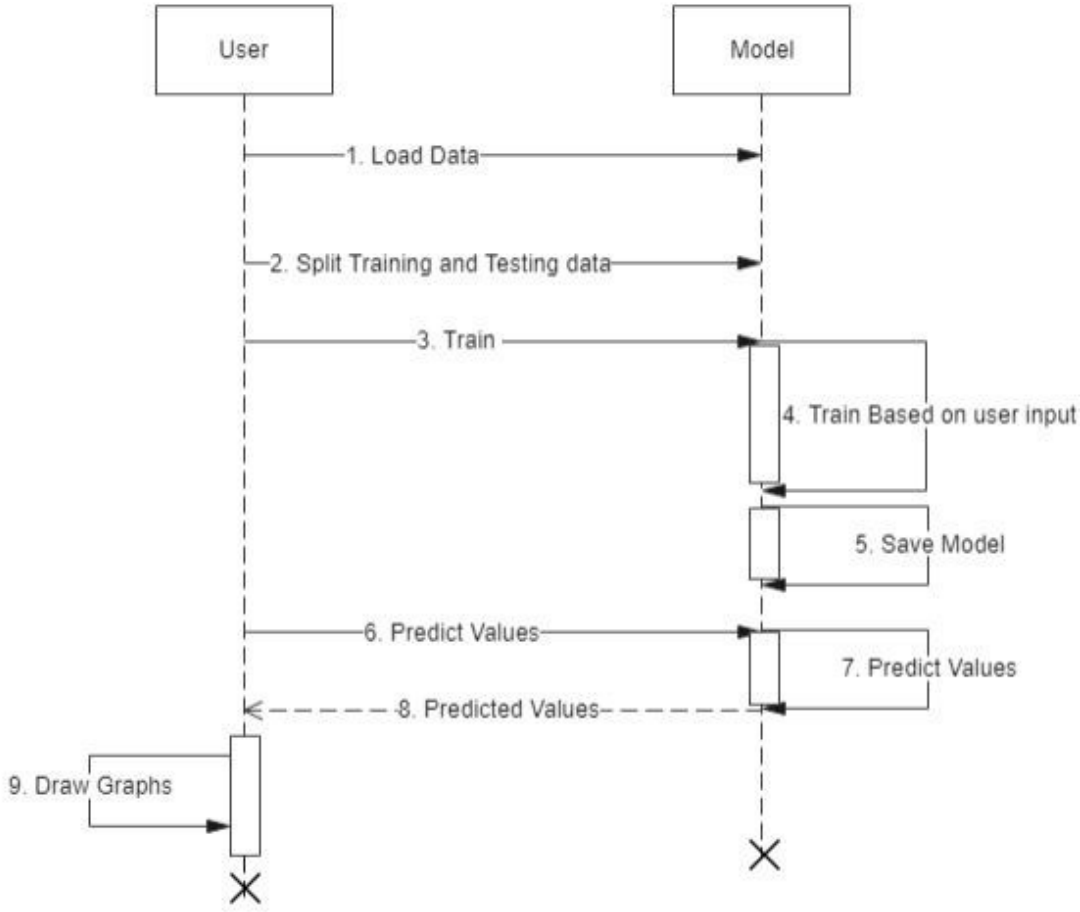


Figure 4.2: Sequence Diagram

The above sequence diagram consists of two entities, user and model. The user initiates the process by loading the data to the model. Next, the user splits the training data and testing data. After splitting the data they train the model. When the model receives train command, it trains the LSTM model using the training data. After training the model, it is saved. Now, the user sends the command to predict the values, upon receiving it, the model

predicts the future values and returns it to the user. Finally the user plots the graphs to derive the outcomes.

4.2.3 Class Diagram

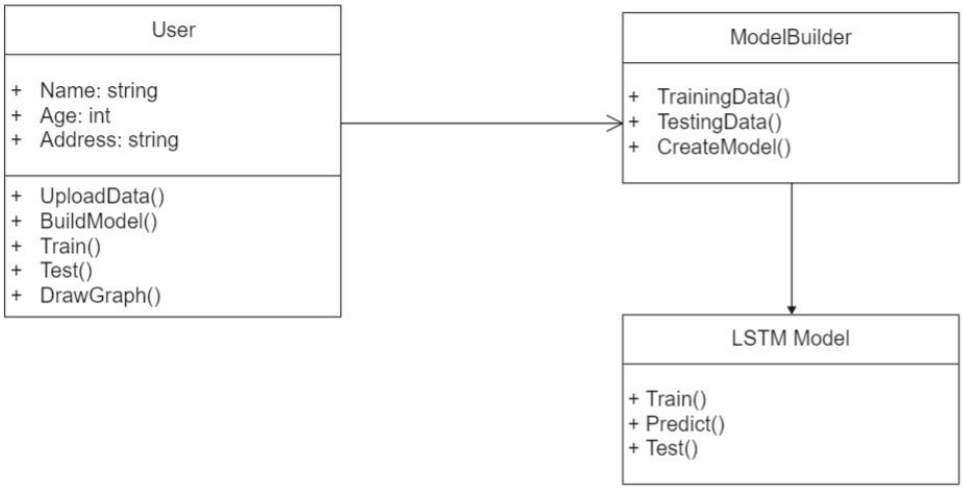


Figure 4.3: Class Diagram

Class diagram is used to specify the various classes that are present the system. In this particular system we have three classes; User, ModelBuilder and LSTM Model. These classes have properties and actions associated to them.

4.2.4 Data Flow Diagram

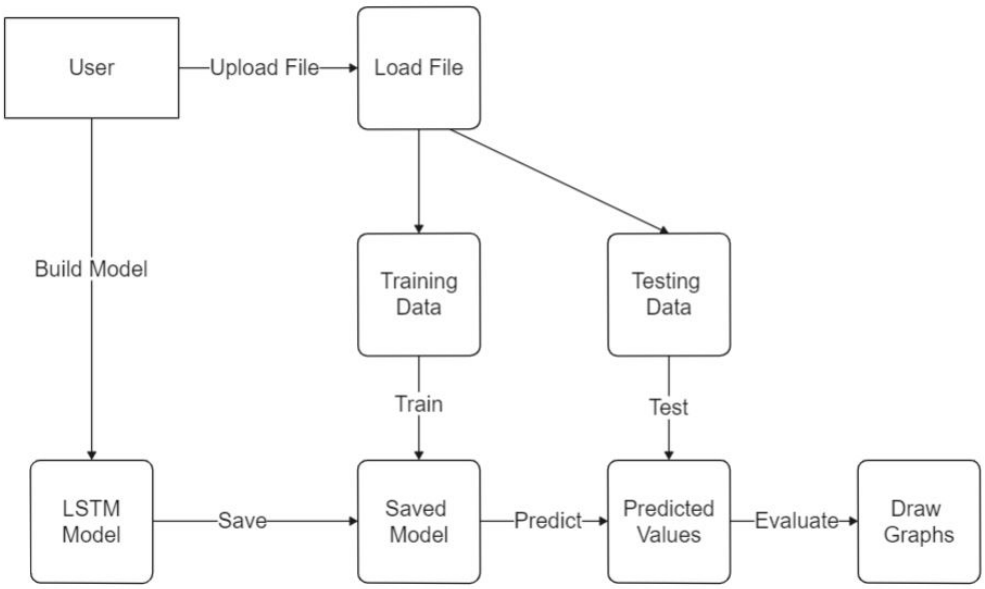


Figure 4.4: Data Flow Diagram

The above data flow diagram is used to illustrate the flow of the data control form the start to the end of the entire process. The user loads the data file, it is then split into training and testing data. Next, the user creates a LSTM model. This model is then saved and trained using the training data. After training it is used to predict the future values and is tested against the testing data. Finally, comparison graphs are plotted.

CHAPTER 5

IMPLEMENTATION AND RESULT ANALYSIS

5.1 Introduction

In order to implement this system, we made use of various tools and dependencies. In this chapter we will be discussing about the steps involved in establishing the tools and the required libraries associated with it. The tool that we made us of is Jupyter Notebook. We will talk about what it is, and how can we run it on our devices. Later on, we will elaborate the implementation procedure that is followed to build the model. Finally, we will be presenting the results that we have obtained on running this system.

5.2 Tools Setup

As mentioned in the above section, we are making use of Jupyter notebook to implement the system. The Jupyter Notebook is an amazingly valuable resource for wisely creating and presenting data science projects.A scratch cushion organizes code and its yield into a single document that joins portrayals, account text, mathematical conditions, and other rich media. Thus: it's a lone record where you can run code, show the yield, and moreover add clarifications, conditions, blueprints, and make your work more clear, justifiable, repeatable, and shareable.

Using Notebooks is as of now a significant piece of the data science work measure at associations across the globe. On the off chance that you will presumably work with data, using a Notebook will speed up your work interaction and simplify it to pass on and share your results.

5.2.1 Installing Jupyter Notebook

The most easy way for a novice in any case Jupyter Notebooks is by presenting Anaconda. It is the most by and large used Python scattering for data science and comes pre-stacked with all of the most notable libraries and gadgets. Without a doubt the best Python libraries associated with Anaconda fuses Numpy, Pandas, Matplotlib, etc. To get Anaconda, basically :

1. Download the most recent form of Anaconda for Python 3.8.
2. Introduce Anaconda by adhering to the directions on the download page or potentially in the executable.

5.2.2 Running Jupyter Notebook

Presently, we need to import the dataset/s that we have accumulated for the ML project nearby. Be that as it may, before we can import the dataset/s, you should set the current registry as the functioning index. You can set the working registry in Jupyter Notebook IDE in three straightforward advances:

1. Save your Python file in the registry containing the dataset.
2. Go to File Explorer choice in Jupyter Notebook IDE and pick the required catalog.
3. Presently, click on the Run alternative to execute the file

5.3 Method of Implementation

The following steps describe the method of implementation of the system:

1. Initially, the dataset was collected from the yahoo finance India website. The data consists of historical prices of bitcoin from the year 2013.
2. Save the dataset in a folder, which has python 3.8 in it, we can also set the environmental variables to access python in any location.
3. Install Anaconda, and other libraries like numpy, pandas, matplotlib, keras and sklearn into this folder.

4. Now, open command prompt with the base directory as this folder. In that type the command: **jupyter notebook**.

5. Now, create a new document with python3 as the type. This generates a notebook in which we can code.

6. Add a cell, which is also referred as code block to write the code.

7. In the very beginning import all the required libraries which will be used in the entire notebook.

8. Now, load the dataset using read_csv() method.

9. Pre-process the dataset, so that all the irregularities and missing values can be omitted.

10. Normalize, the data inorder to avoid the over-fitting and under-fitting problems.

11. After normalizing the data, split them to training set and testing set,in the ratio 3:1

12. Now, initialize a neural network model by using sequential() as the model type.

13. To this model add LSTM layers along with the dropout layers (to avoid over-fitting).

14. Train the above generated model with the help of the training data. After training, save the model.

15. Now, use the trained model to predict the future values and evaluate them against the testing data.

16. Plot the related graphs to gain more insight into the result analysis.

This is process that can be followed to implement the price prediction model using jupyter notebook.

5.4 Result Analysis

During this analysis we have utilized the dataset gathered from the yahoo finance India website. The dataset consists of 2440 instances with 7 attributes each. We have then built the LSTM model by making use of various hyper parameters. We used sigmoid function as the activation function for the LSTM model. We have built the model with 4 LSTM layers having dimensionality of 50 for the output space. The dropout value of 0.2 is used for each layer. Later, when it came to compilation we made use of "Adam" optimizer and the loss functions was MSE (Mean Squared Error). We trained the network for 50 epochs with the batch size of 32. The experiment was conducted on a laptop with core i5 9th Gen processor, 8GB RAM, and Windows 10 OS installed. The model loss is represented in below graph.

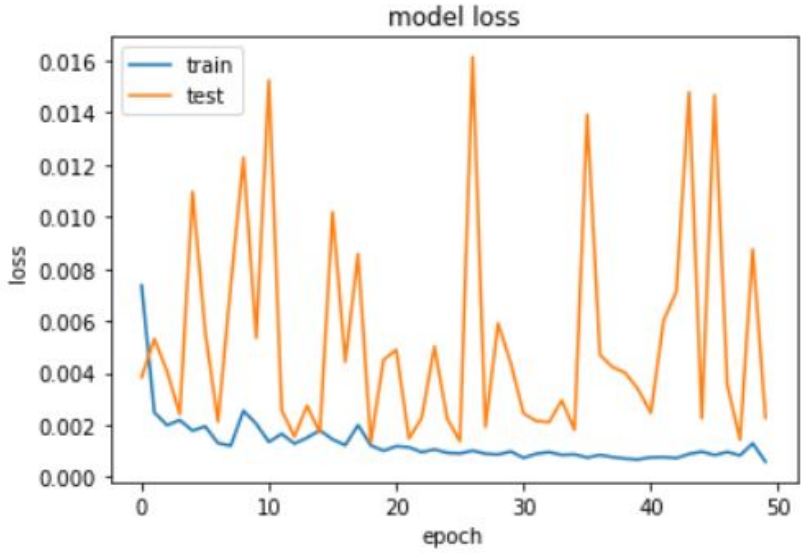


Figure 5.1: Model Loss

The details regarding different error values have been tabulated in the table given below.

MSE	50105501.10
MAE	3403.08
RMSE	7078.52

Table 5.1: Error Values

The above values tells us about the model's efficiency and other parameters. The following graph "Price v/s Dates" tells the price pattern of the bitcoin.

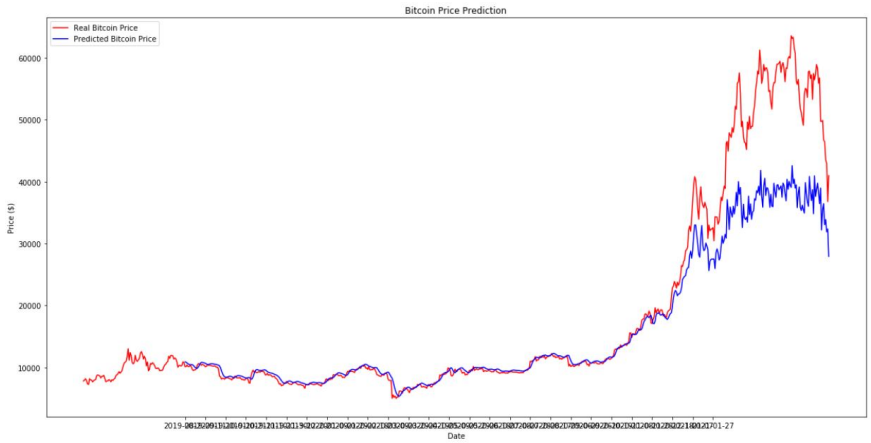


Figure 5.2: Price Pattern

We have not only predicted the regular price but also tried to forecast the price pattern for the next 20 days. The same can be observed in the graph mentioned below.

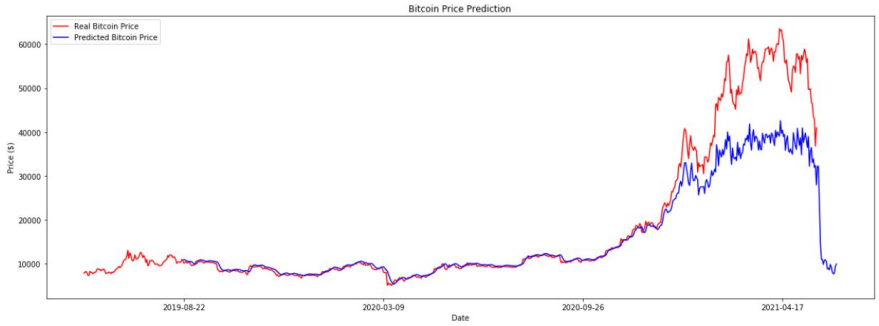


Figure 5.3: Future Price Pattern

CHAPTER 6

TESTING AND VALIDATIONS

6.1 Introduction

Programming testing is a fundamental part of programming quality certification what's more, addresses a definitive review of assurance, plan and coding. The extending detectable quality of programming as a system part and escort costs related with an item dissatisfaction are moving factors for, we organized, through testing. Testing is the route toward executing a program with the assumption for finding a bungle. The arrangement of tests for programming and other planned things can be just probably as trying as the fundamental arrangement of the genuine thing. There of basically three sorts of testing techniques:

- **Black Box Testing:** What's particular about the black-box testing strategy is that the analyzers performing it has no information on the inner design and source code of the product they're trying. Also, they don't have to have any, neither do they need to have inside and out information on programming dialects or remarkable coding abilities to play out the tests. This is essentially on the grounds that the objective of this testing technique isn't to delve profound into the code, going through the product's internal parts, yet to associate with its UI, test its usefulness, and ensure that each information and yield of the framework meets the predetermined prerequisites. Subsequently discovery testing can likewise be called practical testing or determination based testing.

- **White Box Testing:** Rather than black-box testing that centers around usefulness, the objective of the white-box testing technique is to play out the investigation of the interior design of programming and the rationale behind it. White-box testing is along these lines here and there called primary testing or rationale driven testing. This technique is tedious

and expects analyzers to have solid coding abilities, full information on the product they are trying, and admittance to all source code and engineering records, in any case, analyzers won't plan legitimate experiments.

- **Grey Box Testing:** The grey box strategy builds the inclusion of testing procedures by zeroing in on every one of the layers of the product tried paying little mind to its intricacy. While discovery analyzers ensure all is great with interfaces and usefulness, and white-enclose analyzers burrow to the inside design and fix the source code of the product, dim box testing manages both simultaneously in a non-meddling way.

6.2 Testcases and Results

Testing is the subsequent stage after the execution stage. This progression by and large tells about how great the framework is and how proficiently the framework is functioning what's more, how versatile the framework is. In this stage the composed code is executed against a considerable lot of the arbitrarily picked experiments and checked whether the test cases are cleared or not on the off chance that the productivity is less, the code is modified once more what's more, executed and checked agreeing this cycle go on proceeds until the until the most noteworthy proficiency is reached and afterward the product is conveyed. In this stage, we by and large test the framework dependent on various experiments and check whether the framework is working acceptable. The following table describes the various testcases are their success status:

TestcaseID	*Testcase*	*Status*
T01	Data File Loaded	Pass
T02	Pre-processing data	Pass
T03	Removing Noises	Pass
T04	Splitting training data and testing data	Pass
T05	Model Building	Pass
T06	Model Training	Pass
T07	Value Prediction	Pass
T08	Drawing Graphs	Pass

Table 6.1: Analysis based on the Testcases

6.3 Validation

Upon completion of the testcase analysis, we have validated the system and derived the following results:

- The model was able to load the dataset and pre-process it accordingly.
- Data was split into training data and testing data as specified.
- Model generation and training occurred as per the desired requirement.
- Future bitcoin prices were predicted based using the trained model.
- Graphs were plotted to gain detailed understandings of the price pattern.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In this study we have tried to overcome the drawbacks of other models to predict the bitcoin prices and have also tried to forecast the price pattern for the next 20 days. In our model i.e. LSTM model we have made use of multiple layers inorder to predict the price pattern and also tried to rectify the vanishing gradient and exploding gradient problem of the traditional RNN's. The dataset used to do this analysis, was gathered from the yahoo finance website that has information regarding various stocks and other digital currencies.

7.2 Future Scope

This system or model works on the sole principle of predicting the bitcoin price by making use of the LSTM model. We are also forecasting the price pattern for next 20 days using this. The time-series analysis and price prediction is not only limited to the use of LSTM model, but one can also make use of other models and various datasets inorder to do so. The other models that can be used include GRU, Stacked LSTM and so on. As in this research we are focusing only on bitcoin, other cryptocurrency prices can also be predicted by making use of this model. Not only cryptocurrencies but one can also use this model to know about the price patterns of various stocks and digital currencies.

REFERENCES

[1] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (2008).

[2] Richa Kaushal. "Bitcoin: first decentralized payment system." In: *Int J Eng Comput Sci 5.5* (2016), pp. 16514–16517.

[3] et al. Polasik Michal. "Price fluctuations and the use of bitcoin: An empirical inquiry". In: *International Journal of Electronic Commerce 20.1* (2015), pp. 9–49.

[4] et al. Kjærland Frode. "How can bitcoin price fluctuations be explained". In: (2018).

[5] URL: `https://coinmarketcap.com/all/views/all/`.

[6] Xiangxi Jiang. "Bitcoin Price Prediction Based on Deep Learning Methods". In: *Journal of Mathematical Finance. 10. 10.4236/jmf.2020.101009.* (2020), pp. 132–139.

[7] *Bitcoin's Price History.* URL: `https://www.investopedia.com/articles/forex/121815/bitcoins-price-history`.

[8] et al Narayanan Arvind. ". Bitcoin and cryptocurrency technologies: a comprehensive introduction". In: *Princeton University Press* (2016).

[9] T. Guo and N. Antulov-Fantulin. ". Predicting short-term Bitcoin price fluctuations from buy and sell orders". In: (2018).

[10] Tata S. Moser A. Smit Almeida J. ". Bitcoin prediction using ANN. Neural Networks". In: (2015).

[11] D.; Cheng H.; Cheng W.; Jiang G.; Cottrell G.W. Qin Y.; Song. ". A dual-stage attention-based recurrent neural network for time series prediction." In: *26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25* (August 2017), pp. 2627–2633.

[12] Negara E.S. Widyanti Y. Ferdiansyah F. ". BITCOIN-USD trading using SVM to detect the current day's trend in the market." In: *J. Inf. Syst. Inform. 1* (2019), pp. 70–77.